VUTTIPITTAYAMONGKOL, P. and ELYAN, E. 2020. Improved overlap-based undersampling for imbalanced dataset classification with application to epilepsy and Parkinson's disease. *International journal of neural systems* [online], 30(8), article ID 2050043. Available from: https://doi.org/10.1142/S0129065720500434.

Improved overlap-based undersampling for imbalanced dataset classification with application to epilepsy and Parkinson's disease.

VUTTIPITTAYAMONGKOL, P. and ELYAN, E.

2020

International Journal of Neural Systems, Vol. 0, No. 0 (2005) 1–??

Improved Overlap-based Undersampling for Imbalanced Dataset Classification with Application to Epilepsy and Parkinson's Disease

Pattaramon Vuttipittayamongkol, Eyad Elyan*
School of Computing Science and Digital Media
Robert Gordon University, Aberdeen, AB10 7GJ, UK
p.vuttipittayamongkol@rgu.ac.uk; e.elyan@rgu.ac.uk

Classification of imbalanced datasets has attracted substantial research interest over the past decades. Imbalanced datasets are common in several domains such as health, finance, security and others. A wide range of solutions to handle imbalanced datasets focus mainly on the class distribution problem and aim at providing more balanced datasets by means of resampling. However, existing literature shows that class overlap has a higher negative impact on the learning process than class distribution. In this paper, we propose overlap-based undersampling methods for maximizing the visibility of the minority class instances in the overlapping region. This is achieved by the use of soft clustering and the elimination threshold that is adaptable to the overlap degree to identify and eliminate negative instances in the overlapping region. For more accurate clustering and detection of overlapped negative instances, the presence of the minority class at the borderline areas is emphasized by means of oversampling. Extensive experiments using simulated and real-world datasets covering a wide range of imbalance and overlap scenarios including extreme cases were carried out. Results show significant improvement in sensitivity and competitive performance with well-established and state-of-the-art methods.

Keywords: class overlap; imbalanced data; undersampling; classification; adaptive threshold; Fuzzy C-means; epilepsy; Parkinson's disease.

1. Introduction

A dataset with a skewed distribution over its classes is called an imbalanced dataset. For example, an imbalanced dataset may contain 90% of samples from Class I and the remaining 10% from Class II. This situation is common in many real-world problems such as anomaly detection,[1] medical diagnosis,[2] object recognition[3] and business analysis.[4] In such domains, the under-represented class is generally the class of interest. Thus, in this paper, where binary-class problems are focused, we refer to the minority class and the majority class as the positive class and the negative class unless otherwise stated.

Since traditional learning algorithms are generally designed to maximize the overall accuracy,[5] they tend to be biased towards the over-represented class in imbalanced scenarios. Oversampling and undersampling data to obtain better class distributions are commonly used to address this issue. Existing data resampling methods that aim to rebalance data distribution have potential to improve classification results.[6–8] However, many of them do not factor in the problem of class overlap, which occurs when examples from different classes share similar values in features. Class overlap is a major obstacle in classification tasks and often shows a higher negative impact than class imbalance.[9–11] Moreover, the impact of class imbalance is highly dependent on the presence of class overlap.[12] This explains better performance of some overlap-based solutions over those that solely

*Corresponding author

focus on rebalancing class distributions.[9,12,13]

A recent method, *Overlap-Based Undersampling (OBU)*, proved to enhance the classification of imbalanced datasets.[14] It aims at maximizing the visibility of minority class instances by eliminating majority class instances in the overlapping region. The authors hypothesized that each class possessed its own uniqueness, and hence the two classes could roughly be represented by two distinct clusters. Any samples with high similarity to the other class' properties were then considered to be in the overlapping region. They employed a soft clustering algorithm to discover such instances, which had indistinctive membership degrees. Results showed competitive performance with a state-of-the-art method and significant improvement in sensitivity. However, some key limitations of the method, such as an empirical setting of the elimination threshold and excessive elimination of majority class instances, need to be addressed.

In this paper, we propose methods that extend OBU to overcome its drawbacks and improve performance. The main contributions are outlined below:

- *Adaptive-threshold OBU (AdaOBU)* is presented with an automatic elimination threshold adaptable to the degree of class overlap for more accurate identification and elimination of overlapped negative instances.
- *Boosted OBU (BoostOBU)* is proposed to reduce excessive elimination of negative instances. This is achieved by improving the performance of the clustering algorithm by means of emphasizing the presence of minority class instances along the borderline. Consequently, the identification and elimination of overlapped majority class instances are more accurate and the excessive elimination is reduced. This is illustrated in Fig 1, where Fig 1(a) shows the original data and Fig 1(b) is the result of OBU. Fig 1(c) shows how the minority class borderline is emphasized leading to less excessive elimination of negative instances as can be seen in Fig 1(d).
- Extensive experiments using simulated and real-world datasets with various learning algorithms were carried out. These cover extreme scenarios of class overlap and class imbalance.
- The proposed methods were successfully applied to automated prediction of Epilepsy and Parkinson's disease.

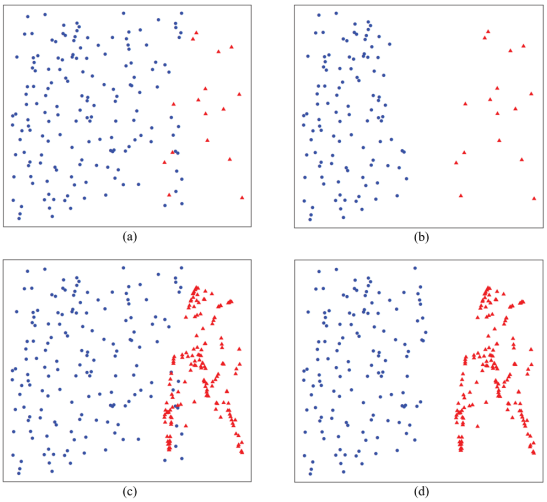

Figure 1. (a) The original data (b) with OBU applied, and (c) the minority class borderline is emphasized before (d) identifying and removing overlapped majority class instances.

2. Related Work

Data resampling is a common approach for handling imbalanced datasets. Many resampling methods mainly focused on rebalancing the dataset.[8,15] However, rebalancing solutions were often outperformed by other methods that mainly focused on the problem of class overlap. This can be explained by several studies that reported a higher negative impact of class overlap on the performance of learning algorithms than class imbalance.[10,11] It was shown that a dataset with no class overlap could be perfectly classified regardless of imbalance degree. Moreover, when the class overlap was low, class imbalance had no significant effects on the classification results.

To tackle the problem of class overlap, methods to resample borderline instances or overlapped instances were proposed. Borderline-SMOTE (BLSMOTE)[16] used neighborhood searching to identify borderline minority class instances, from which new instances were synthesized. DBMUTE[13] employed a density-based clustering algorithm to locate instances in different areas, and undersampled majority class instances in the overlapping area. It was shown that BLSMOTE and DBMUTE performed better than Safe-level-SMOTE[17] (SLSMOTE) and DBSMOTE,[7] which utilized the same algorithms as them to identify different types of instances but avoided resampling in the overlapping region. Since DBMUTE does not balance class distributions, this

evidences a higher impact of class overlap.

In SMOTE-IPF,[18] noisy instances were removed after performing SMOTE.[6] By doing so, sparse positive instances near the borderline mistaken as noise would no longer appear as noise and hence would not be filtered out. Also, by having more positive instances in the overlapping area, some negative instances were filtered out, which enhanced the visibility of the positive class to the learning algorithm.

In Support Vector Machine (SVM), support vectors are mostly composed of borderline instances while non-support vectors reside further away from the borderline. Based on this premise, the authors of Ref. 19 proposed to oversample and undersample support and non-support vectors, respectively. By doing so, more informative instances were emphasized and information loss could be minimized.

To the best of our knowledge, more of overlap-based methods considered borderline instances and relatively few have been proposed to address the entire overlapping region. To maximize the visibility of the minority class, neighborhood-based (NB-based) undersampling[12] was proposed to remove majority class instances from the overlapping region. Four methods based on neighborhood searching to accurately locate overlapped majority class instances were designed. Competitive results with other state-of-the-art methods were achieved. Similarly, ADASYN enhanced the presence of the minority class by oversampling in the overlapping region and showed improvement in sensitivity.[20] However, as opposed to NB-based undersampling, ADASYN does not guarantee the maximum visibility of the positive class instances as negative instances will still be present in the overlapping region.

A redundancy-driven method was proposed to progressively eliminate overlapped negative instances based on similarity and contribution factors.[21] Even though class overlap was considered, the elimination was carried out until the classes were balanced. Thus, applying this method on a highly imbalanced dataset could result in excessive elimination of negative instances, and on the other hand, insufficient removal may occur when imbalance is low.

HardEnsemble incorporated oversampling and undersampling to address overlapped instances of both classes.[22] Undersampling was based on instance's contribution to the classification accuracy, which potentially facilitated removal of majority class instances in the overlapping region. Under the same criterion, oversampling was done particularly on overlapped minority class instances. These processes were carried out simultaneously and the resulting datasets were used to train RUSBoost.[23] Another ensemble-based method utilizing an Evolutionary Algorithm (EA) was proposed.[24] EA was used to selectively remove overlapped majority class instances. These ensemble-based methods often showed better performance than other state-of-the-art methods. However, they require high computational complexities, especially when EA is also used.

3. Methods

3.1. *Related algorithms*

3.1.1. *FCM: Fuzzy C-means*

Fuzzy C-means (FCM)[25] is one of the most commonly-used soft clustering algorithms. Unlike hard clustering, a soft clustering algorithm allows each instance to be a member of many clusters. The likelihood of belonging to a cluster is expressed as a membership degree, whose value is between 0 and 1. The membership degrees of an instance sum up to 1. FCM follows a similar procedure to the k-means algorithm. It starts by randomly initializing cluster centroids. Then, the within-cluster variance is calculated from the fractional distances of all instances to each centroid as expressed in Eq. 1, where μ_{ij} is the membership degree of x_i in the cluster j, x_i is the i^{th} instance, and c_j is the cluster centroid. Subsequently, the new centroids are recalculated. These steps are iterated until the objective function is minimized.

$$J_m = \sum_{i=1}^{N}\sum_{j=1}^{C} \mu_{ij}^{l} ||x_i - c_j||^2 \quad , \quad 1 \leq m \leq \infty \quad (1)$$

3.1.2. *OBU: Overlap-Based Undersampling*

OBU aims at maximizing the visibility of the positive class by removing most negative instances in the overlapping region.[14] The authors proposed that in a binary-class dataset, the classes could mainly be represented by two unique clusters based on their outstanding characteristics and thus samples in the overlapping region would have high similarity to the other cluster's properties. They employed FCM to discover the two unique clusters and identify samples that potentially resided in the overlapping region.

Algorithm 1: OBU Algorithm

input : traning set $T = T_{neg} \cup T_{pos}$, elimination threshold μ_{th}
output : resampled training set

```
begin
    T ← FCM(T, cluster = 2)
    T_neg_new ← subset(T_neg, x_i | μ_ineg ≥ μ_th)
    T_OBU ← T_neg_new ∪ T_pos
end
```

As described in Alg. 1, FCM is applied to the training set T to determine two distinct clusters and assign membership degrees to each sample indicating the likelihood of belonging to the two clusters. Negative instances that have high membership degrees in the positive cluster are considered potentially overlap with positive instances, and thus are eliminated from the training set. The cut-off membership degree for potential overlapped instances, i.e. the elimination threshold (μ_{th}), needs to be empirically set.

3.1.3. *BLSMOTE: Borderline-SMOTE*

BLSMOTE is an improvement of SMOTE[6] by oversampling only borderline samples.[16] It is based on the idea that samples far from the borderline are less likely to be misclassified and hence contribute less to the classification. In BLSMOTE, minority class samples are identified as "*danger*", "*safe*" and "*noise*" based on the number of majority class samples in their k nearest neighbors. Only *danger* samples, which are highly likely to be in the borderline region, are used for linear-interpolating oversampling. BLSMOTE has two models – BLSMOTE1 and BLSMOTE2. BLSMOTE1 only generates new instances from the *danger* samples and their minority class nearest neighbors whereas in BLSMOTE2 all nearest neighbors are considered regardless of class.

3.2. *The proposed algorithms*

3.2.1. *AdaOBU: Adaptive-threshold OBU*

AdaOBU incorporates an adaptive elimination threshold in OBU as shown in Fig. 2 allowing the method to be more generalized across datasets with varying overlap degrees. The adaptive threshold is self-adjusting to the fuzziness of the dataset, which is indicated by how similar instances are to their own class' properties on average. By this definition, it is suggested that a dataset is fuzzy when a large number of instances have indistinct membership degrees. In other words, we can say that there likely to be a high class overlap when a dataset is highly fuzzy.

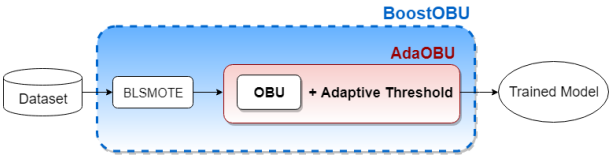

Figure 2. A diagram showing the extensions of OBU by AdaOBU and BoostOBU

The algorithm of AdaOBU is shown in Alg. 2. Line $3-5$ express how the adaptive threshold (μ_{th}) is computed. First, the average membership degrees of all negative instances belonging to the negative cluster ($\bar{\mu}_{neg}$) and the positive cluster ($\bar{\mu}_{pos}$) are calculated. Then, the minimum between $\bar{\mu}_{neg}$ and $\bar{\mu}_{pos}$ is used as μ_{th}. The rationale behind this is as follows. The difference between the two means ($|\bar{\mu}_{neg} - \bar{\mu}_{pos}|$) indicates the fuzziness of the dataset. According to FCM, membership degrees range between 0 and 1; thus, $0 \leq |\bar{\mu}_{neg} - \bar{\mu}_{pos}| \leq 1$. In an extreme case when $|\bar{\mu}_{neg} - \bar{\mu}_{pos}|| = 0$, none of the clusters shows distinct nature of the negative class. This suggests high overlapping between the two classes. The opposite applies in the other extreme case of $|\bar{\mu}_{neg} - \bar{\mu}_{pos}| = 1$. Accordingly, the overlapping degree and hence elimination amount are to be proportional to the smaller value between $\bar{\mu}_{neg}$ and $\bar{\mu}_{pos}$. Then, the elimination process as of Line $6-7$ of Alg. 2 is followed.

Algorithm 2: AdaOBU Algorithm

input : traning set $T = T_{neg} \cup T_{pos}$
output : resampled training set

```
1 begin
2     T ← FCM(T, cluster = 2)
3     μ̄_neg ← mean(μ_ineg | x_i ∈ T_neg)
4     μ̄_pos ← mean(μ_ipos | x_i ∈ T_neg)
5     μ_th ← min(μ̄_neg, μ̄_pos)
6     T_neg_new ← subset(T_neg, x_i | μ_ineg ≥ μ_th)
7     T_AdaOBU ← T_neg_new ∪ T_pos
8 end
```

3.2.2. *BoostOBU: Boosted OBU*

BoostOBU is a hybrid method presented to improve the detection of negative instances in the overlapping region, hence reducing excessive elimination. Erroneous elimination of OBU could have been partly due to low visibility of positive instances within the overlapping region, which caused poor performance of the clustering algorithm. To address this issue, BoostOBU aims at improving the presence of the

positive class, especially along the borderline, before applying clustering. The adaptive elimination threshold proposed in 3.2.1 is also integrated in BoostOBU as illustrated in Fig. 2.

Algorithm 3: BoostOBU Algorithm

input : traning set $T = T_{neg} \cup T_{pos}$
output : resampled training set

begin
  $(T_{BS} = T_{neg} \cup T_{pos\_new}) \leftarrow BLSMOTE(T)$
  $T \leftarrow FCM(T_{BS}, cluster = 2)$
  $\bar{\mu}_{neg} \leftarrow mean(\mu_{ineg} | x_i \in T_{neg})$
  $\bar{\mu}_{pos} \leftarrow mean(\mu_{ipos} | x_i \in T_{neg})$
  $\mu_{th} \leftarrow min(\bar{\mu}_{neg}, \bar{\mu}_{pos})$
  $T_{neg\_new} \leftarrow subset(T_{neg}, x_i | \mu_{ineg} \geq \mu_{th})$
  $T_{BoostOBU} \leftarrow T_{neg\_new} \cup T_{pos\_new}$
end

For the purpose of emphasizing the border of the minority class, BLSMOTE1 is used. This can be justified as follows. Firstly, BLSMOTE proved to successfully improve the visibility of minority class borders to the learning algorithm with higher sensitivity over SMOTE[16] achieved. Secondly, since it avoids oversampling noisy samples, the effect of noise would not be enlarged. Thirdly, BLSMOTE1 only synthesizes based on minority class samples, thus it is ensured that the minority class border is highlighted rather than being expanded.

Alg. 3 outlines BoostOBU. BLSMOTE is first applied and followed by overlap-based undersampling based on the adaptive elimination threshold.

4. Experiments

Extensive experiments covering a wide range of imbalanced and overlapped datasets were carried out. This includes 66 synthetic datasets and 68 public real-world datasets, 2 of which are large and high-dimensional. Results were compared against well-established methods and state-of-the-art methods. The Friedman test and 1xN post-hoc Wilcoxon signed rank tests with Holm correction were carried out to assess the significance of the result improvement. For reproducibility, the code of our methods and simulated datasets used is available on *GitHub*[a].

4.1. *Setup*

Three sets of experiments were carried out. Simulated datasets and small to medium-sized real-world datasets were used in Experiment I and Experiment II, respectively. Experiment III was carried out on large real-world datasets. SVM was chosen as the learning algorithm. Sensitivity, specificity, G-mean, and F1-score were used to asses the methods. Results were compared against SMOTE,[6] BLSMOTE,[16] k-means undersampling (kmUnder)[8] and OBU.[14] In Experiment II, further evaluation using Decision Tree (J48), k-nearest neighbors (kNN) and Random Forests (RF), and comparison against more robust methods, SMOTE-ENN,[26] SMOTEBagging[27] (SMTBagging) and RUSBoost[23] were carried out.

4.2. *Overlap quantification*

Since class overlap has not been mathematically well-characterised, several approaches have been formulated to estimate the overlap degree.[9,12,28] However, these methods are not generalised across datasets due to some constrains such as the requirement of normal distribution of data[28] and inconsistency in results.[9] In Ref. 12, the authors suggested that the overlap degree of simulated data should be determined with respect to the minority class. This is because the minority class is naturally more overwhelmed by the class overlap as depicted in Fig. 3.

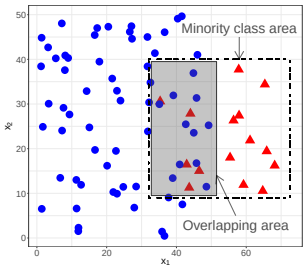

Figure 3. Area approximations

For the purpose of synthesising various overlap scenarios in our experiment, we followed the method of Ref. 12 expressed as in Eq. 2, where the area approximations are illustrated in Fig. 3.

$$overlap(\%) = \frac{overlapping\ area}{minority\ class\ area} * 100 \qquad (2)$$

4.3. *Datasets*

In Experiment I, we used 66 simulated binary-class datasets,[12] which cover a wide range of class over-

[a]https://github.com/fonkafon/BoostedOBU

lap and class imbalance degrees. The class imbalance degree was calculated as $imb = \frac{N}{P}$, where N and P are the numbers of negative and positive instances. To evaluate the performance of our methods in relation to the changes in class imbalance and class overlap, the datasets were uniformly distributed in two-dimensional space (*i.e.* data densities of the positive and negative classes were equal in each dataset).

All datasets were generated with a fixed number of negative samples and fixed data space of the positive class. The number of positive samples was based on the imbalance degree. The density of the negative class was made equal to that of the positive class. Thus, at a higher imbalance degree, data density was lower. This resulted in many variations of datasets. Fig. 4 illustrates two examples of simulated datasets. Both datasets have 100 negative samples. There are 6 positive samples in Fig. 4(a) and 33 in Fig. 4(b) making $imb = 15$ and $imb = 3$. The density of data in Fig. 4(a) is lower than that in Fig. 4(b) (axes are of different scales).

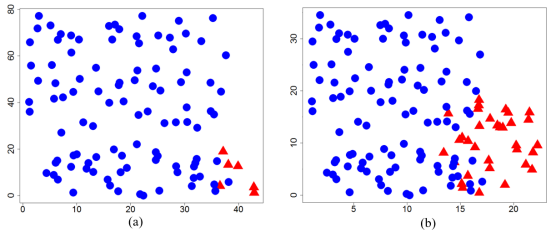

Figure 4. Examples of two synthetic datasets with 50% class overlap and (a) $imb = 15$ and (b) $imb = 3$.

For Experiment I, we simulated datasets with the imbalance degrees of $1.5, 3, 12, 30, 60$ and 120. At each imbalance degree, the overlap degree ranged from $0\% - 100\%$ in a step of 10. The number of negative instances generated in each dataset was $6,000$ while the number of positive instances was varied between $50 - 4,000$ based on the imbalance degree.

In Experiment II, 66 datasets from *UCI Repository*[29] and *KEEL Repository*[30] were used. As shown in Table 1, the datasets vary in imbalance degrees (*1.82-129.44*), number of features (*3-34*), and number of instances (*92-5,472*).

Experiment III was carried out on large and high-dimensional datasets. These were the breast cancer dataset from *KDD Cup 2008*[b] and the handwritten digits dataset from *the MNIST database*.[31] The breast cancer dataset is binary-class with 117 features and 102,294 samples. It contains 101,671 negative and 623 positive samples, which makes $imb = 163.20$. The handwritten digits dataset is 10-class with 784 features and 60,000 samples. As our methods are designed for binary-class datasets, we used the one-vs-all scheme to make two binary-class datasets, MNIST_3 and MNIST_5, which had class 3 and class 5 as the minority class. These two classes were ones of hard-to-classify numbers and even the most challenging classes for a deep-learning based method.[32] In each dataset, the minority class was undersampled to obtain a higher imbalance degree. In MNIST_3, class 3 was undersampled such that $imb = 43.90$, which made of 53,869 negative and 1,227 positive instances. In MNIST_5, class 5 was undersampled such that $imb = 20.13$, which made of 53,869 negative and 2,711 positive instances.

For all datasets, partitioning of 80:20 of training to testing sets was used. To diminish the effect of noisy instances, we normalized data using standard scores. In Experiment I and II, 10-fold cross-validation was employed in the training phase for the purpose of model selection. No cross-validation was applied to the large datasets in Experiment III as sufficient training data was available.

Table 1. Datasets used in Experiment II

Dataset	Instances	Imb	f
Glass1	214	1.82	9
Ecoli0vs1	220	1.86	7
Wisconsin	683	1.86	9
Pima	768	1.87	8
Iris0	150	2	4
Glass0	214	2.06	9
Yeast1	1484	2.46	8
Haberman	306	2.78	3
Vehicle2	846	2.88	18
Vehicle1	846	2.9	18
Vehicle3	846	2.99	18
Glass0123vs456	214	3.2	9
Vehicle0	846	3.25	18
Ecoli1	336	3.36	7
Newthyroid1	215	5.14	5
Newthyroid2	215	5.14	5
Ecoli2	336	5.46	7
Segmemt0	2308	6.02	19
Glass6	214	6.38	9
Yeast3	1484	8.1	8
Ecoli3	336	8.6	7
Pageblocks0	5472	8.79	10
Yeast2vs4	514	9.08	8
ecoli067vs35	222	9.09	7
glass015vs2	172	9.12	9
yeast02579vs368	1004	9.14	8
ecoli046vs5	203	9.15	6
ecoli0267vs35	224	9.18	7
glass04vs5	92	9.22	9
ecoli0346vs5	205	9.25	7
Yeast05679vs4	528	9.35	8
Vowel0	988	9.98	13
ecoli067vs5	220	10	6

Dataset	Instances	Imb	f
ecoli0147vs2356	336	10.59	7
led7digit02456789vs1	443	10.97	7
ecoli01vs5	240	11	6
glass0146vs2	205	11.06	9
Glass2	214	11.59	9
cleveland0vs4	177	12.62	13
ecoli0146vs5	280	13	6
Shuttle0vs4	1829	13.87	9
Yeast1vs7	459	14.3	7
Glass4	214	15.46	9
Ecoli4	336	15.8	7
Pageblocks13vs2	472	15.86	10
Abalone0918	731	16.4	8
dermatology6	358	16.9	34
Glass016vs5	184	19.44	9
Shuttle2vs4	129	20.5	9
Yeast1458vs7	693	22.1	8
Glass5	214	22.78	9
Yeast2vs8	482	23.1	8
Yeast4	1484	28.1	8
winequalityred4	1599	29.17	11
Yeast1289vs7	947	30.57	8
winequalityred8vs6	656	35.44	11
Ecoli0137vs26	281	39.14	7
abalone21vs8	581	40.5	8
Yeast6	1484	41.4	8
winequalitywhite3vs7	900	44	11
winequalityred8vs67	855	46.5	11
abalone19vs10111213	1622	49.69	8
winequalitywhite39vs5	1482	58.28	11
shuttle2vs5	3316	66.67	9
winequalityred3vs5	691	68.1	11
Abalone19	4174	129.44	8

[b]https://www.kdd.org/kdd-cup/view/kdd-cup-2008

4.4. *Parameter Settings*

To provide a fair comparison, no parameters tuning or optimization were performed in our experiments. AdaOBU has no free parameters. In BoostOBU, the k value of BLSMOTE was set to 5. For SMOTE, BLSMOTE, OBU, and SMOTE-ENN the same parameter settings as in the original work were used. These were $k = 5$ in SMOTE and BLSMOTE, and $\mu_{th} = 0.45$ in OBU. In SMOTE-ENN, $k = 5$ and $k = 3$ were set for SMOTE and ENN. As for SMT-Bagging and RUSBoost, 40 and 10 weak learners were used, respectively, as suggested by Ref. 33.

The Radial Bias Function kernel was used for SVMs with the default settings of $cost\,(C) = 1$ and $\gamma = \frac{1}{f}$, where f is the number of features. For J48 and RF, the number of features determined at each split, $mtry = \sqrt{f}$. The number of trees ($mtree$) in RF was 500. Lastly, $k = 5$ was used for kNN.

5. Results and Discussion

5.1. *Experiment I: Simulated datasets*

Results on 66 simulated datasets are shown in Fig. 5. The performance of OBU and the proposed extensions are marked with dashed lines, and the other methods are marked with solid lines. The shaded areas are the areas under the performance curves of the baseline (SVM with no resampling).

BoostOBU achieved the top performance across all metrics in most imbalance and overlap scenarios. AdaOBU showed competitive performance with OBU across all metrics and provided comparable results with well-established and state-of-the-art methods, especially at higher imbalance degrees.

In Fig. 5, AdaOBU showed clear improvement over the baseline in sensitivity and G-mean across most imbalance and overlap degrees. This is also confirmed by its average performance across 66 scenarios given in Table 2, where the top result in each metric is highlighted in bold. The symbols next to each value indicate the results of significance tests at 95% confidence level comparing the results cross 66 datasets. An asterisk (*) denotes a statistically significant difference between the method and AdaOBU, and a dagger (†) denotes a statistically significant difference between the method and BoostOBU.

Table 2 shows that AdaOBU improved sensitivity from 67.75% to 97.5% and G-mean from 77.86% to 91% on average. As can be seen in Fig. 5, AdaOBU was competitive in sensitivity, specificity and G-mean with SMOTE, BLSMOTE and kmUnder at higher imbalance degrees. However, AdaOBU suffered from high false positives (FP), especially when the imbalance and overlap degrees were high. This must have been caused by excessive elimination as a result of inaccurate identification and removal of negative instances. More excessive elimination was likely to occur at higher imbalance and overlap degrees, where the visibility of the minority class to the clustering algorithm was lower, resulting in poorer performance of the clustering algorithm. Only in a few cases with no overlap or slight overlap that AdaOBU showed the smallest FP compared to the other methods since a smaller number of negative instances was removed. As shown in Table 2, AdaOBU achieved higher F1-score and had competitive sensitivity, specificity and G-mean with OBU on average indicating less excessive elimination. Therefore, the proposed adaptive threshold has shown to be able to effectively replace the free parameter in OBU.

From Table 2, BoostOBU achieved the highest average sensitivity (99.5%) and F1-score (77.73%). Even though the average specificity of BoostOBU was not as high as SMOTE, BLSMOTE and kmUnder, Fig. 5 shows that BoostOBU, in fact, provided competitive specificity with those methods across most imbalance degrees. The exception occurred at very low imbalance degrees, especially at $imb = 1.5$, where BoostOBU outperformed the other methods in sensitivity but suffered from low specificity. Similarly, at all imbalance and overlap levels, except at $imb = 1.5$, BoostOBU often achieved the highest G-mean among all methods. This indicates a good trade-off between the accuracy of the positive and the negative classes achieved by BoostOBU. In terms of F1-score, BoostOBU performed competitively with SMOTE, BLSMOTE and kmUnder. However, Fig. 5 shows that BoostOBU clearly outperformed these methods at very high to extreme imbalance degrees, *i.e.* $imb = 60$ and 120. These results indicate that BoostOBU not only could provide the highest sensitivity but also significantly reduced the number of false positives from OBU.

In conclusion, the competitive and higher results of AdaOBU compared to OBU across a wide range of overlap and imbalance scenarios proved that the proposed adaptive threshold could potentially replace parameter tuning in OBU. The significantly better

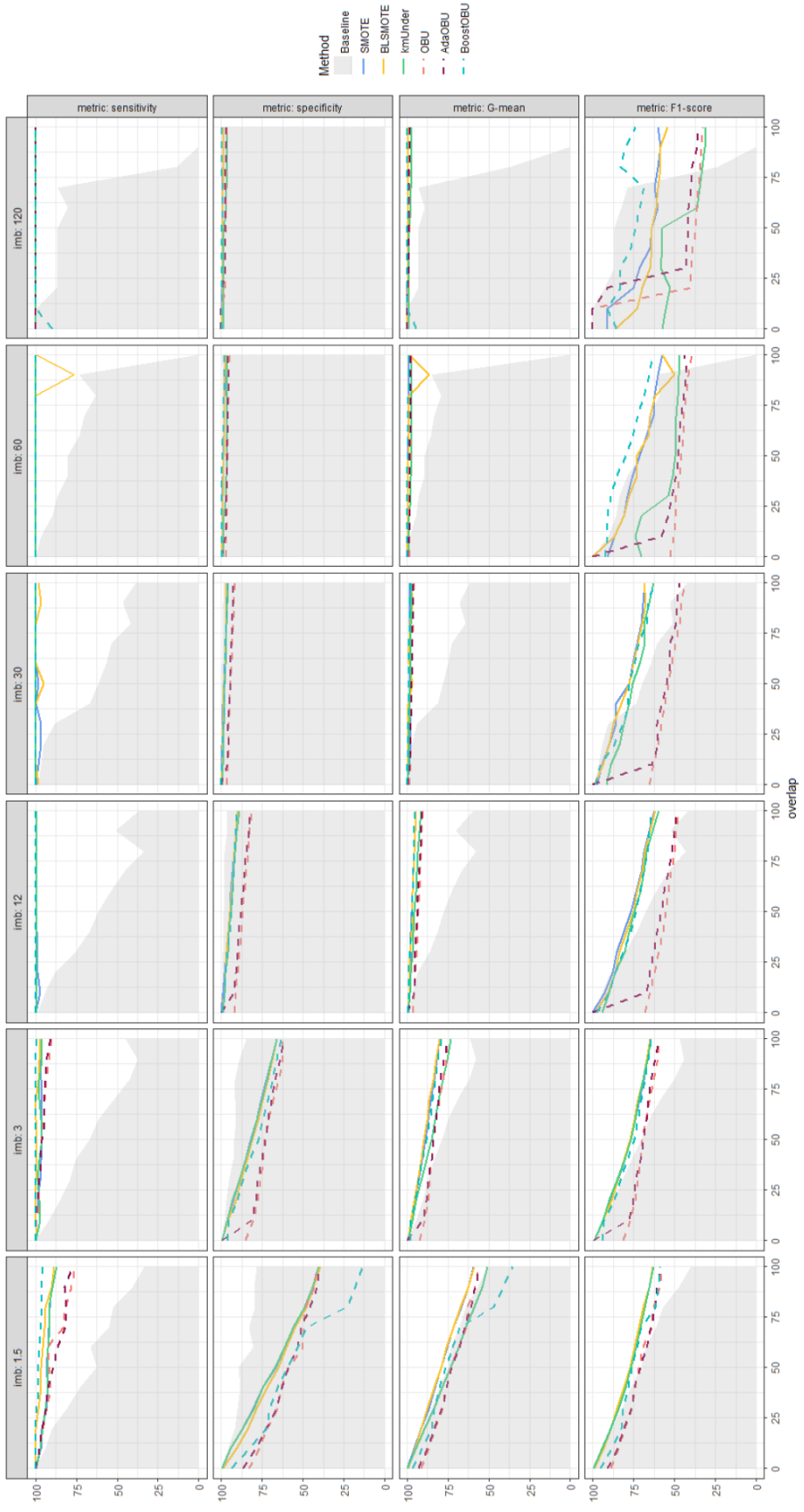

Figure 5. Performance of the methods in terms of sensitivity, specificity, G-mean and F1-score across various imbalance and overlap degrees in Experiment I, where each column of the subplots shows the results on a specific imbalance degree of data with the full range of class overlap.

Table 2. Average results and statistical test results from Experiment I

	Baseline	SMOTE	BLSMOTE	kmUnder	OBU	AdaOBU	BoostOBU
sensitivity	67.75*†	98.11†	98.56†	98.35†	97.46†	97.5†	**99.5**
specificity	96.2*†	**90.35***†	89.87*†	89.63*	84.38*†	85.41†	87.47
G-mean	77.86*†	**93.87***	93.78*	91.71†	90.43*†	91†	92.41
F1-score	69.88*†	76.41*	75.59*	68.54*†	57.57*†	62.04†	**77.73**

*The difference in results of the method and of AdaOBU is statistically significant.
†The difference in results of the method and of BoostOBU is statistically significant.

performance of BoostOBU over OBU and AdaOBU across all metrics (Table 2) suggests that emphasizing the presence of borderline positive instances helped improve the detection of overlapped negative instances. Moreover, BoostOBU outperformed all of the well-established and state-of-the-art methods in sensitivity and F1-score while achieving competitive performance in specificity and G-mean. These results show that BoostOBU provided the most optimized solution among the methods.

5.1.1. *Adaptive threshold analysis*

We collected the threshold values for analysing its relation to imbalance and overlap degrees. Results verified that the adaptive threshold was automatically proportional to the amount of overlapped samples.

Fig. 6 presents the plots of μ_{th} in different scenarios. Across all imbalance degrees, the plots show a clear trend of μ_{th} increasing with the degree of class overlap as hypothesized. From no overlap to complete overlap, the variations in μ_{th} were 12.41% at $imb = 1.5$, 12.34% at $imb = 3$, 8.88% at $imb = 12$, 3.59% at $imb = 30$, 5.07% at $imb = 60$ and 1.82% at $imb = 120$. This result shows that the proposed adaptive threshold was able to adapt to changes in the overlap degree making the elimination amount directly proportional to the degree of class overlap.

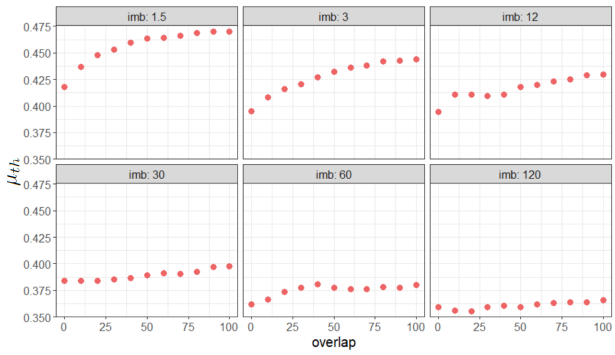

Figure 6. Self-adaptability of the elimination threshold at different imbalance and overlap degrees

Fig. 6 also shows that μ_{th} is inversely proportional to the imbalance degree. As discussed in Section 4.3, at a higher imbalance degree, there were fewer negative instances in the overlapping region. Consequently, fewer negative instances would be removed. This is another evidence that μ_{th} was able to self-adjust to different overlap scenarios.

5.2. *Experiment II: Real-world datasets*

The performance of AdaOBU and BoostOBU on 66 real-world datasets was consistent with that in Experiment I, apart from slight variations in the ranks, which was partly due to more comparison methods added in this experiment. AdaOBU and BoostOBU were among the methods that provided highest sensitivity. Their G-mean and F1-score were also comparable with others. Due to space limit, Table 3-6 show results of 24 representative datasets sorted from low to high imbalance degrees as examples to aid the discussion. These 24 examples were selected to cover all ranges of imbalance ratios and all performance behaviors of 66 datasets. However, the discussion will be based on the results of the 66 datasets, whose detailed results are available on *GitHub*.

In Table 3-6, ranks based on the performance compared across all methods are also provided next to the metric values. Rank 1 indicates the best performance across all methods on the dataset, and so on. The average rank of each method and significance test results across all 66 datasets are provided.

As seen in Table 3, AdaOBU achieved the highest sensitivity rank followed by OBU, kmUnder and BoostOBU. AdaOBU provided the highest sensitivity on 41 datasets and Boost OBU on 31 datasets. More importantly, both methods outperformed the ensemble-based methods, namely SMTBagging and RUSBoost. In particular, AdaOBU was significantly better than SMTBagging as well as the baseline, SMOTE, BLSMOTE and SMOTE-ENN in sensitivity. The imbalance degree did not seem to affect the performance of AdaOBU and BoostOBU, which was

Table 3. Sensitivity results from Experiment II

Dataset	Sensitivity Value/Rank																	
	Baseline		SMOTE		BLSMOTE		kmUnder		SMOTE-ENN		SMTBagging		RUSBoost		OBU		AdaOBU		BoostOBU	
Glass1	25	9	37.5	7	62.5	4	62.5	4	12.5	10	37.5	7	62.5	4	75	2	**87.5**	**1**	75	2
Glass0	80	8	90	3	60	10	90	3	80	8	90	3	90	3	90	3	**100**	**1**	**100**	**1**
Vehicle1	**100**	**1**	**100**	**1**	**100**	**1**	**100**	**1**	**100**	**1**	90	8	**100**	**1**	**100**	**1**	90	8	90	8
Vehicle3	0	7	0	7	33.33	3	33.33	6	33.33	3	33.33	3	0	7	**100**	**1**	66.67	2	0	7
Vehicle0	66.67	8	60	9	80	5	73.33	7	60	9	80	5	86.67	4	**100**	**1**	**100**	**1**	**100**	**1**
Ecoli2	80	4	**100**	**1**	80	4	60	9	60	9	80	4	**100**	**1**	80	4	**100**	**1**	80	4
Segmemt0	18.75	10	50	6	43.75	8	56.25	4	31.25	9	56.25	4	50	6	75	2	68.75	3	**81.25**	**1**
Pageblocks0	81.98	10	92.79	5	95.5	3	91.89	6	93.69	4	91.89	7	90.09	8	87.39	9	**100**	**1**	**100**	**1**
glass015vs2	0	9	33.33	4	33.33	4	**66.67**	**1**	0	9	33.33	4	33.33	4	**66.67**	**1**	**66.67**	**1**	33.33	4
Vowel0	89.23	9	96.92	6	89.23	9	**100**	**1**	98.46	5	95.38	8	95.38	7	**100**	**1**	**100**	**1**	**100**	**1**
cleveland0vs4	50	2	0	6	0	6	**100**	**1**	0	6	0	6	50	2	50	2	50	2	0	6
Yeast1vs7	41.86	9	88.37	5	81.4	8	95.35	2	25.58	10	88.37	5	95.35	2	**95.35**	**1**	93.02	4	88.37	5
Ecoli4	38.1	9	71.43	7	76.19	6	85.71	3	21.43	10	69.05	8	78.57	5	85.71	2	**90.48**	**1**	83.33	4
Shuttle2vs4	41.18	10	74.12	6	78.82	4	71.76	7	51.76	9	71.76	8	75.29	5	82.35	3	85.88	2	**87.06**	**1**
Yeast2vs8	0	9	**33.33**	**1**	**33.33**	**1**	33.33	6	0	9	**33.33**	**1**	16.67	7	**33.33**	**1**	16.67	7	**33.33**	**1**
Yeast4	70	9	90	2	**100**	**1**	90	2	40	10	80	7	90	2	80	7	90	2	90	2
Yeast1289vs7	75	3	75	3	0	8	50	6	0	8	75	3	**100**	**1**	0	8	**100**	**1**	25	7
winequalityred8vs6	0	10	33.33	8	**100**	**1**	**100**	**1**	66.67	7	33.33	8	**100**	**1**	66.67	4	66.67	4	66.67	4
Yeast6	62.5	9	**100**	**1**	87.5	6	75	7	50	10	**100**	**1**	**100**	**1**	75	7	**100**	**1**	**100**	**1**
winequalityred8vs67	0	6	0	6	0	6	**100**	**1**	0	6	0	6	33.33	5	**100**	**1**	**100**	**1**	66.67	4
abalone19vs10111213	0	10	16.67	4	16.67	4	33.33	3	16.67	8	16.67	4	16.67	8	**50**	**1**	**50**	**1**	16.67	4
winequalitywhite39vs5	0	7	0	7	20	6	**100**	**1**	0	7	0	7	60	2	40	3	40	3	40	3
winequalityred3vs5	0	10	50	2	50	2	**100**	**1**	50	2	50	2	50	2	50	2	50	2	50	2
Abalone19	71.43	8	85.71	3	85.71	3	71.43	9	42.86	10	85.71	3	**100**	**1**	85.71	3	**100**	**1**	85.71	3
Average‡		5.64*†		3.73*		4.82*†		2.77		5.88*†		3.91*		3.17		2.12		**2.09**†		2.91

‡The average and the significance test results are based on 66 datasets.

Table 4. Specificity results from Experiment II

Dataset	Specificity Value/Rank																	
	Baseline		SMOTE		BLSMOTE		kmUnder		SMOTE-ENN		SMTBagging		RUSBoost		OBU		AdaOBU		BoostOBU	
Glass1	**99.27**	**1**	91.24	4	86.13	6	84.67	7	98.54	2	91.97	3	81.75	8	67.88	9	61.31	10	87.59	5
Glass0	**100**	**1**	**100**	**1**	92.86	7	98.21	5	**100**	**1**	**100**	**1**	96.43	6	57.14	9	55.36	10	67.86	8
Vehicle1	**100**	**1**	96.88	6	**100**	**1**	93.75	7	90.63	9	**100**	**1**	**100**	**1**	93.75	7	**100**	**1**	59.38	10
Vehicle3	**100**	**1**	94.29	5	94.29	5	57.14	8	97.14	3	94.29	5	**100**	**1**	31.43	9	28.57	10	97.14	3
Vehicle0	**88.89**	**1**	81.48	2	70.37	5	66.67	6	77.78	3	77.78	3	59.26	7	25.93	10	51.85	8	51.85	9
Ecoli2	**100**	**1**	**100**	**1**	**100**	**1**	**100**	**1**	**100**	**1**	**100**	**1**	94.59	10	**100**	**1**	**100**	**1**	**100**	**1**
Segmemt0	**100**	**1**	57.78	6	55.56	8	66.67	3	71.11	2	62.22	4	55.56	7	53.33	9	57.78	5	33.33	10
Pageblocks0	**98.17**	**1**	94.2	7	91.65	8	95.21	4	96.54	3	94.6	5	94.4	6	98.07	2	23.01	10	36.56	9
glass015vs2	**100**	**1**	96.77	4	80.65	6	54.84	8	**100**	**1**	**100**	**1**	90.32	5	16.13	10	29.03	9	64.52	7
Vowel0	96.96	5	99.75	2	96.96	5	99.24	4	**100**	**1**	99.49	3	94.94	7	64.3	9	61.77	10	81.52	8
cleveland0vs4	**100**	**1**	**100**	**1**	**100**	**1**	65.63	10	**100**	**1**	**100**	**1**	**100**	**1**	**100**	**1**	**100**	**1**	**100**	**1**
Yeast1vs7	**93.6**	**1**	72	6	70.4	7	76	4	91.2	2	78.4	3	68	8	21.6	10	24	9	72.8	5
Ecoli4	**94.44**	**1**	76.19	5	76.19	5	74.6	7	**94.44**	**1**	79.37	3	76.98	4	22.22	10	27.78	9	32.54	8
Shuttle2vs4	**93.36**	**1**	72.99	5	61.61	7	70.62	6	85.31	2	74.88	3	72.99	4	56.87	8	56.87	9	51.18	10
Yeast2vs8	**100**	**1**	80.3	7	80.3	7	75.76	10	98.48	2	83.33	5	90.91	4	83.33	5	78.79	9	93.18	3
Yeast4	**98.91**	**1**	95.65	4	95.65	4	93.48	7	**98.91**	**1**	95.65	4	92.39	8	88.04	9	86.96	10	97.83	3
Yeast1289vs7	**100**	**1**	91.3	8	94.57	7	72.83	9	97.83	5	95.65	6	**100**	**1**	**100**	**1**	71.74	10	98.91	4
winequalityred8vs6	**100**	**1**	92.13	3	94.49	2	38.58	10	91.34	5	92.13	3	79.53	7	70.87	9	78.74	8	89.76	6
Yeast6	99.31	2	98.61	4	97.92	5	94.1	9	99.31	2	97.92	5	94.79	8	95.14	7	90.97	10	**99.65**	**1**
winequalityred8vs67	**100**	**1**	93.41	5	94.61	4	38.92	10	93.41	6	96.41	3	78.44	7	73.05	8	67.07	9	97.01	2
abalone19vs10111213	**100**	**1**	89.94	3	83.33	7	66.04	10	92.14	2	87.42	5	89.62	4	66.35	8	66.35	8	84.91	6
winequalitywhite39vs5	**99.31**	**1**	92.78	5	90.72	6	11.68	10	93.47	3	97.59	2	89	7	81.44	8	80.76	9	93.13	4
winequalityred3vs5	**100**	**1**	96.32	5	97.79	3	50	10	96.32	4	96.32	5	88.24	7	63.97	8	58.82	9	**100**	**1**
Abalone19	**100**	**1**	94.46	5	88.24	8	94.12	6	**100**	**1**	91.35	7	84.43	10	95.5	4	86.51	9	95.85	3
Average		**1.51***†		3.52*†		4.8*		6.29		2.23*†		3.09*†		5.74*		6.38†		6.69†		4.37

consistent with the results in Experiment I.

Non-winning cases in sensitivity of AdaOBU may have been due to other variations such as data density that we have not considered in this work. In most cases that AdaOBU improved the sensitivity over OBU, highest sensitivity was achieved. There were few exceptions, for example, on Shuttle2vs4, where BoostOBU improved the performance further from AdaOBU and had the highest sensitivity. The decreases in sensitivity on Vehicle1, Vehicle3, Yeast1vs7 and Yeast2vs8 from OBU evidence unsuccessful cases of the adaptive threshold. Since the adaptive threshold is solely distance-based, other factors such as local data density may have caused the inaccuracy during the clustering process. Similarly, the results on cleveland0vs4, Yeast4, winequalityred8vs6 and some others where none of the OBU-based methods won suggested that considering only the distance factor may not be sufficient. Many non-winning cases of BoostOBU over AdaOBU such as Vehicle3, Yeast1289vs7 and abalone19vs10111213 were highly likely affected by the poor performance of BLSMOTE as can be seen in Table 3.

Table 4 shows that all methods commonly led to decreases in specificity. These were due to the trade-offs for higher sensitivity, except for SMOTE-ENN,

Table 5. G-mean results from Experiment II

Dataset	G-mean Value/Rank																			
	Baseline		SMOTE		BLSMOTE		kmUnder		SMOTE-ENN		SMTBagging		RUSBoost		OBU		AdaOBU		BoostOBU	
Glass1	49.82	9	58.49	8	73.37	2	72.75	4	35.1	10	58.73	7	71.48	5	71.35	6	73.25	3	**81.05**	**1**
Glass0	89.44	5	**94.87**	**1**	74.64	8	94.02	3	89.44	5	**94.87**	**1**	93.16	4	71.71	10	74.4	9	82.38	7
Vehicle1	**100**	**1**	98.43	4	**100**	**1**	96.82	5	95.2	7	94.87	8	**100**	**1**	96.82	5	94.87	8	73.1	10
Vehicle3	0	7	0	7	56.06	2	43.64	6	**56.9**	**1**	56.06	2	0	7	56.06	2	43.64	5	0	7
Vehicle0	76.98	2	69.92	8	75.03	3	69.92	7	68.31	9	**78.88**	**1**	71.66	6	50.92	10	72.01	4	72.01	5
Ecoli2	89.44	4	**100**	**1**	89.44	4	77.46	9	77.46	9	89.44	4	97.26	3	89.44	4	**100**	**1**	89.44	4
Segmemt0	43.3	10	53.75	5	49.3	8	61.24	3	47.14	9	59.16	4	52.7	6	**63.25**	**1**	63.03	2	52.04	7
Pageblocks0	89.71	8	93.49	4	93.55	2	93.54	3	**95.11**	**1**	93.24	5	92.22	7	92.57	6	47.97	10	60.46	9
glass015vs2	0	9	56.8	3	51.85	5	**60.46**	**1**	0	9	57.74	2	54.87	4	32.79	8	43.99	7	46.37	6
Vowel0	93.02	6	98.32	3	93.02	6	**99.62**	**1**	99.23	2	97.42	4	95.16	5	80.19	9	78.6	10	90.29	8
cleveland0vs4	70.71	2	0	6	0	6	**81.01**	**1**	0	6	0	6	70.71	2	70.71	2	70.71	2	0	6
Yeast1vs7	62.6	7	79.77	5	75.7	6	**85.13**	**1**	48.3	8	83.24	2	80.52	3	45.38	10	47.25	9	80.21	4
Ecoli4	59.98	6	73.77	5	76.19	3	**79.97**	**1**	44.99	9	74.03	4	77.77	2	43.64	10	50.13	8	52.07	7
Shuttle2vs4	62	10	73.55	2	69.69	6	71.19	4	66.45	9	73.31	3	**74.13**	**1**	68.44	7	69.89	5	66.75	8
Yeast2vs8	0	9	51.74	4	51.74	4	50.25	6	0	9	52.7	2	38.92	7	52.7	2	36.24	8	**55.73**	**1**
Yeast4	83.21	9	92.78	3	**97.8**	**1**	91.72	4	62.9	10	87.48	7	91.19	5	83.93	8	88.47	6	93.83	2
Yeast1289vs7	86.6	2	82.75	5	0	8	60.34	6	0	8	84.7	3	**100**	**1**	0	8	84.7	4	49.73	7
winequalityred8vs6	0	10	55.42	8	**97.21**	**1**	62.11	7	78.03	3	55.42	8	89.18	2	68.73	6	72.45	5	77.36	4
Yeast6	78.78	9	99.3	2	92.56	6	84.01	8	70.46	10	98.95	3	97.36	4	84.47	7	**95.38**	**5**	99.83	1
winequalityred8vs67	0	6	0	6	0	6	62.39	4	0	6	0	6	51.13	5	**85.47**	**1**	81.89	2	80.42	3
abalone19vs10111213	0	10	38.72	5	37.27	9	46.92	3	39.19	4	38.17	7	38.65	6	**57.6**	**1**	**57.6**	**1**	37.62	8
winequalitywhite39vs5	0	7	0	7	42.6	5	34.18	6	0	7	0	7	**73.08**	**1**	57.08	3	56.84	4	61.03	2
winequalityred3vs5	0	10	69.4	5	69.93	3	**70.71**	**1**	69.4	4	69.4	5	66.42	7	56.56	8	54.23	9	**70.71**	**1**
Abalone19	84.52	8	89.98	5	86.97	7	81.99	9	65.47	10	88.49	6	91.89	2	90.48	4	**93.01**	**1**	90.64	3
Average		5.3		4.03		5.48		4.08		5.73		**3.97**		4.26		4.89		4.68		4.48

Table 6. F1-score results from Experiment II

Dataset	F1-score Value/Rank																			
	Baseline		SMOTE		BLSMOTE		kmUnder		SMOTE-ENN		SMTBagging		RUSBoost		OBU		AdaOBU		BoostOBU	
Glass1	36.36	2	26.09	7	31.25	3	29.41	4	18.18	10	27.27	5	26.32	6	20.69	8	20.59	9	**38.71**	**1**
Glass0	88.89	4	**94.74**	**1**	60	7	90	3	88.89	4	**94.74**	**1**	85.71	6	41.86	10	44.44	9	52.63	8
Vehicle1	**100**	**1**	95.24	4	**100**	**1**	90.91	8	86.96	9	94.74	5	**100**	**1**	90.91	7	94.74	6	56.25	10
Vehicle3	0	7	0	7	33.33	2	10.53	6	**40**	**1**	33.33	2	0	7	20	4	13.33	5	0	7
Vehicle0	71.43	2	62.07	8	68.57	5	62.86	7	60	9	**72.73**	**1**	66.67	6	60	9	69.77	3	69.77	4
Ecoli2	88.89	3	**100**	**1**	88.89	3	75	9	75	9	88.89	3	83.33	8	88.89	3	**100**	**1**	88.89	3
Segmemt0	31.58	9	37.21	6	32.56	8	45	3	29.41	10	42.86	5	36.36	7	**48.98**	**1**	47.83	2	44.07	4
Pageblocks0	82.73	3	76.01	6	70.9	8	78.46	4	83.53	2	76.69	5	75.19	7	**85.46**	**1**	22.7	10	26.27	9
glass015vs2	0	9	40	2	20	5	21.05	4	0	9	**50**	**1**	28.57	3	12.9	8	14.81	6	13.33	7
Vowel0	85.93	5	97.67	3	85.93	5	97.74	2	**99.22**	**1**	96.12	4	84.35	7	47.97	9	46.26	10	64.04	8
cleveland0vs4	**66.67**	**1**	0	6	0	6	26.67	5	0	6	0	6	**66.67**	**1**	**66.67**	**1**	**66.67**	**1**	0	6
Yeast1vs7	52.17	7	65.52	5	60.87	6	**71.93**	**1**	33.85	10	70.37	2	66.13	3	45.05	8	44.94	9	66.09	4
Ecoli4	49.23	6	58.82	5	61.54	3	**65.45**	**1**	31.03	10	59.79	4	63.46	2	40.91	9	44.44	7	43.21	8
Shuttle2vs4	52.24	10	61.46	2	57.51	6	58.65	4	55	9	61.31	3	**62.14**	**1**	56.91	7	58.63	5	56.49	8
Yeast2vs8	0	9	11.76	4	11.76	4	10	7	0	9	13.33	2	10.53	6	13.33	2	5.71	8	**23.53**	**1**
Yeast4	77.78	4	78.26	3	83.33	2	72	6	53.33	10	72.73	5	69.23	7	55.17	9	58.06	8	**85.71**	**1**
Yeast1289vs7	85.71	2	40	4	0	8	12.9	7	0	8	54.55	3	**100**	**1**	0	8	23.53	6	33.33	5
winequalityred8vs6	0	10	14.29	5	**46.15**	**1**	7.14	9	25	2	14.29	5	18.75	4	9.52	8	12.5	7	22.22	3
Yeast6	66.67	4	80	2	66.67	4	38.71	9	57.14	6	72.73	3	51.61	7	42.86	8	38.1	10	**94.12**	**1**
winequalityred8vs67	0	6	0	6	0	6	5.56	4	0	6	0	6	5	5	11.76	2	9.84	3	**40**	**1**
abalone19vs10111213	0	10	5.13	4	3.33	9	3.45	8	**6.25**	**1**	4.26	6	5	5	5.17	2	5.17	2	3.64	7
winequalitywhite39vs5	0	7	0	7	6.06	5	3.75	6	0	7	0	7	**15**	**1**	6.56	3	6.35	4	14.81	2
winequalityred3vs5	0	10	25	3	33.33	2	5.56	7	25	3	25	3	10.53	6	3.85	8	3.39	9	**66.67**	**1**
Abalone19	**83.33**	**1**	41.38	5	25.53	9	34.48	6	60	2	31.58	7	23.73	10	46.15	4	26.42	8	48	3
Average		4*		3.55*		5.36		4.97		4.98		**3.48***		4.91		5.62†		5.7†		4.17

which had poorer performance than the baseline in both sensitivity and specificity. AdaOBU, which achieved the highest average rank in sensitivity, had the lowest specificity on average. This indicates that the trade-offs of AdaOBU were high, which may not be suitable for some application domains.

Comparing BoostOBU with the other methods, its winning over BLSMOTE and RUSBoost on both sensitivity and specificity proved that BoostOBU provided a better solution than these approaches. The method also showed higher average specificity than kmUnder while their average sensitivity ranks were comparable. Compared with OBU and AdaOBU, which had higher sensitivity, BoostOBU had significantly higher average specificity. In contrast, its specificity was lower than SMOTE, SMOTE-ENN and SMTBagging, which achieved lower sensitivity. However, BoostOBU won on 24 out of 66 datasets whereas each of SMOTE and SMTBagging only had 21 winning cases. These results only suggest different trade-offs between sensitivity and specificity of BoostOBU and these methods. Their g-mean and F1-score will be discussed for a more conclusive comparison.

Table 5 shows that AdaOBU and BoostOBU had higher average G-mean than the baseline,

BLSMOTE, SMOTE-ENN and OBU. However, the statistical tests did not indicate significant differences between our methods and the others. Thus, it may be said that AdaOBU and BoostOBU had comparable G-mean to the other methods on average. However, it is worth pointing out that AdaOBU and BoostOBU achieved the highest G-mean on 18 and 20 datasets while SMTBagging and RUSBoost, which had higher average ranks, only won on 16 and 15 datasets.

In Table 6, BoostOBU showed significantly higher average rank on F1-score than OBU and AdaOBU. This suggests that BoostOBU provided a better trade-off between the accuracy of the two classes than OBU and AdaOBU. Even though BoostOBU had a lower average rank than SMTBagging and SMOTE, it far outnumbered the two methods in winning cases by 23 to 17 and 18, respectively. Extremely low F1-score can be observed in Table 6, especially on large and highly imbalanced datasets. In many cases, e.g. on Yeast6 and Abalone19, low F1-score is seen although high values in the other metrics were achieved. This is because F1-score factors in precision, which considers true positives (TP) and FP. On a large and highly imbalanced scenario, the calculation of F1-score can be heavily dominated by high FP regardless of specificity. The 23 winning cases of BoostOBU were spread throughout all imbalance degrees. In particular, it handled extremely imbalanced datasets better than the the other methods. This is evidence that BoostOBU performed the best among the methods in minimising information loss while maximising sensitivity.

Both AdaOBU and BoostOBU have shown their superior results over other well-established and state-of-the-art methods including ensemble-based methods in many cases. AdaOBU achieved the highest average sensitivity but suffered from high information loss in the negative class. BoostOBU, which often provided high sensitivity and most favourable trade-offs of relatively smaller FP, may be more preferred in many problem domains.

For further evaluation, an additional experiment using J48, kNN and RF was carried out (Detailed results are available on *GitHub*). Statistical analysis suggests that there were no significant differences in the results using SVM compared to J48 and RF. However, AdaOBU with kNN performed poorer than AdaOBU with SVM across all metrics. Our results also showed that BoostOBU with kNN achieved significantly higher sensitivity and lower performance in other metrics compared to SVM. These results are consistent with literature,[34] which showed that SVM outperformed other algorithms in sensitivity when there were fewer negative instances in the overlapping region.

5.3. *Experiment III: Large datasets*

Table 7 shows the results on the three large and high-dimensional datasets. In all scenarios, AdaOBU obtained higher sensitivity than OBU, and BoostOBU further improved from AdaOBU. Results in other measures varied across datasets.

On the breast cancer dataset, AdaOBU and BoostOBU significantly improved sensitivity from the baseline, BLSMOTE, and OBU. They outperformed kmUnder in specificity, and outperformed the baseline and OBU in G-mean. BoostOBU also achieved higher G-mean than BLSMOTE. AdaOBU and BoostOBU suffered from high FP as can be seen from low F1-score. It is worth pointing out that none of the methods could yield high sensitivity without a high decrease in F1-score. This trade-off was likely caused by the issue of high class overlap. This is evidenced by the results of SMOTE, which showed significant improvement in sensitivity from 28.23% to 70.16% and slightly lower specificity from 99.96% to 97.5% compared to the baseline. However, F1-score of SMOTE was largely reduced from 41.92% to 24.17% due to the bias caused by relatively large FP compared to the number of TP.

On MNIST_3, BoostOBU was among the methods that produced the most favorable results. BoostOBU showed good performance across all metrics. It achieved the second-highest sensitivity of 92.24%, high specificity of 99.13%, the highest G-mean of 95.62%, and relatively high F1-score of 80%. This was significantly higher than the overall performance of kmUnder, which produced the highest sensitivity but very low specificity, G-mean and F1-score. AdaOBU showed improvement over OBU in sensitivity and G-mean, however suffered from high FP. BoostOBU improved further from AdaOBU with higher sensitivity and a reduction in FP as reflected by high specificity and F1-score. Note that OBU with the fixed elimination threshold used in Ref. 14 failed to undersample this dataset as well as MNIST_5.

On MNIST_5, AdaOBU and BoostOBU

Table 7. Results on the large datasets from Experiment III

Dataset	Metric	Baseline	SMOTE	BLSMOTE	kmUnder	OBU	AdaOBU	BoostOBU
Breast Cancer	sensitivity	28.23	70.16	55.65	**94.35**	42.74	58.87	75
	specificity	99.96	97.5	97.08	66.34	**99.91**	78.37	78.83
	G-mean	53.12	**82.71**	73.5	79.12	65.35	67.92	76.89
	F1-score	41.92	24.17	17.56	3.3	**54.08**	3.18	4.11
MNIST_3	sensitivity	82.45	87.76	84.08	**95.92**	82.45	87.35	92.24
	specificity	99.8	99.82	**99.89**	36.56	99.8	97.75	99.13
	G-mean	90.71	93.6	91.64	59.22	90.71	92.4	**95.62**
	F1-score	86.14	**89.77**	88.98	6.43	86.14	61.06	80
MNIST_5	sensitivity	90.96	93.91	93.91	**95.94**	90.96	93.73	94.1
	specificity	99.61	99.61	**99.69**	91.81	99.61	85.95	95.38
	G-mean	95.18	96.72	**96.76**	93.85	95.18	89.75	94.74
	F1-score	91.47	93.05	**93.82**	53.17	91.47	39.32	65.55

provided competitive sensitivity with SMOTE, BLSMOTE and kmUnder and outperformed the baseline and OBU. AdaOBU did not performed as well as the other methods in terms of specificity, G-mean and F1-score due to excessive elimination. Consequently, BoostOBU showed low F1-score. However, BoostOBU had reasonable specificity and G-mean, and produce higher specificity, G-mean and F1-score than kmUnder.

The proposed AdaOBU and BoostOBU performed relatively well on the large datasets in terms of sensitivity compared to other methods. Competitive results in specificity and G-mean were achieved in some cases. However, they often suffered from high FP partly due to the trade-off nature on a large and highly imbalanced datasets.

5.4. *Discussion*

Results on simulated and real-world datasets showed that our proposed methods often achieved high sensitivity. Compared to other methods, BoostOBU, in particular, provided higher sensitivity with better trade-offs of relatively smaller FP in most cases. The improvement in sensitivity of our methods is attributed to better visibility of the minority class near the borderline, which was obtained after removing majority class instances from the overlapping region. This allowed the learning algorithm to learn the maximum boundary of the minority class without interference of majority class instances. Moreover, by removing majority class instances in the overlapping region, the effect of class imbalance is also reduced. This helps improve the results in the case that the proposed methods fail to completely remove class overlap. Furthermore, by oversampling borderline minority class instances in BoostOBU to enhance the performance of the clustering algorithm, the presence of the minority class near the boundary was also increased as an additional benefit.

Higher sensitivity and better trade-offs of BoostOBU over other methods can be justified as follows. While BoostOBU attempted to maximize the presence of the minority class near the borderline, SMOTE and k-means undersampling only aimed to rebalance the class distribution. The improvement in the presence of the minority class in the overlapping region by SMOTE and k-means undersampling was limited by the imbalance degree. Also, as opposed to k-means undersampling, BoostOBU was unlikely to remove instances outside of the overlapping region causing smaller unnecessary information loss. Lastly, BLSMOTE only dealt with borderline instances whereas BoostOBU addressed the entire overlapping region. These enabled BoostOBU to achieve higher sensitivity and higher F1-score. This higher F1 score can be attributed to a higher increase in TP in relation to a smaller FP, which indicates a good trade-off of the method.

6. Applications

This section demonstrates the used of the proposed methods in predictive diagnostics of neurological disorders that widely affect people around the world and increase the risk of premature death – epilepsy and Parkinson's disease.

6.1. *Epileptic seizure*

Epilepsy is a neurological disorder that causes unprovoked, recurrent seizures due to abnormal electrical activity in the brain. There are many types of seizures depending on which part of the brain is affected and how far it spreads. Some types cause patients to lose awareness, which can lead to serious physical injuries while some can cause sudden un-

explained death.[35] Electroencephalogram (EEG) is a common test used in diagnosing epilepsy.[36] Several methods for computer-aided detection of epileptic seizures based on EEG have been proposed.[35,36] Many of these employed artificial neural network algorithms for more accurate prediction.[37,38] Deep learning algorithms often provide promising results; however, they require a sufficiently large number of training samples and are computationally expensive. Also, among these methods, the problem of class imbalance in epilepsy data was rarely discussed.

We propose the use of our resampling methods on an epileptic seizure problem. We used an epileptic seizure recognition dataset from UCI repository[29] containing 11,500 samples, of which are 2,300 epileptic seizure (positive) cases and 9,200 cases with no seizures (negative). Each sample contains 178 features, which are the values of EEG recorded at a different point in time. The same experimental settings as in Section 4 were used with SVM as the baseline.

Results in Table 8 showed that BoostOBU achieved the highest detection rate of epileptic seizures of 98.26% with the geometric mean of the correctly predicted positive and negative cases comparable to other methods of 95.53%. The results among BoostOBU, SMOTE, BLSMOTE, SMOTE-ENN, SMTBagging and RUSBoost were competitive in sensitivity and G-mean, which were better than kmUnder, OBU and AdaOBU. It can be observed that all methods provided high accuracy. This could be partly due to the availability of sufficient samples of both classes and a low imbalance degree even though there is a high degree of class overlap as evidenced by μ_{th} near 0.5 of AdaOBU and BoostOBU.

6.2. *Parkinson's disease*

Parkinson's Disease (PD) is a progressive nervous system disorder that affects movement. It is the second most common neurological disease, which impacts a large number of elders worldwide.[39] PD gradually damages the brain, thus early diagnosis will be beneficial in treatment to reduce symptoms. However, clinical symptoms are not noticeable in an early stage.[40] Mathematical-related and computer-assitsted methods have been proposed to address to issue.[40–43] Existing approaches were based on detecting potential PD cases from EEG,[42] gait or speech signals[39,41] and images.[43]

Here, we demonstrate the use of our proposed method to classify PD samples and healthy samples based on a speech dataset. The dataset obtained from UCI repository[29] contains 195 samples, which are 147 samples with PD and 48 healthy samples, and each sample has 23 features. Note that on this dataset, even though the positive class (PD) is the majority class, all resampling methods were not modified and were applied based on the minority and majority class concept.

Results in Table 8 showed 100% accuracy of BoostOBU on both PD and healthy test cases, which clearly outperformed all other methods. AdaOBU could increase the prediction accuracy of healthy cases to 100% however with the decreased accuracy on PD samples of 89.66%. The low μ_{th} indicates that the dataset tends to have small class overlap; thus, and relatively fewer majority class samples might have been removed by BoostOBU. With more accuracy and more necessary removal of overlapped majority class instances performed by BoostOBU compared to other methods, BoostOBU was able to achieve the highest performance.

7. Conclusions

In this paper, new overlap-based undersampling methods were proposed. By removing negative instances from the overlapping region based on an adaptive threshold, exceptional improvement in minority class accuracy with relatively small trade-offs of false positives was achieved. The proposed methods proved to enhance the classification of well-known imbalanced datasets and outperformed existing methods across a wide range of simulated and real-world datasets of varying class imbalance and class overlap degrees. These results can be attributed to some advantages of our methods over other common undersampling approaches. First, the adaptive elimination threshold enables the undersampling amount to be proportional to the overlap degree. This also results in minimizing the excessive elimination, which reduces information loss. Second, enhancing the presence of the positive class near the borderline areas showed to be beneficial to the overall performance of the method.

Future work will address limitations of the methods. These may include the dependencies on the techniques used such as BLSMOTE and the distance-based algorithms. Results showed that the performance of BoostOBU could be highly depen-

Table 8. Results on Epilepsy and Parkinson's Predictions

Dataset	Metric/μ_{th}	Baseline	SMOTE	BLSMOTE	kmUnder	SMOTE-ENN	SMTBagging	RusBoost	OBU	AdaOBU	BoostOBU
Epilepsy	sensitivity	92.83	97.83	96.52	93.04	97.83	95.87	98.04	92.83	96.96	**98.26**
	specificity	98.1	97.23	97.45	95.63	97.28	**98.15**	97.12	98.1	85.92	92.88
	G-mean	95.43	97.53	96.98	94.33	97.55	97	**97.58**	95.43	91.27	95.53
	F1-score	92.62	93.65	93.38	**98.21**	93.75	94.33	93.57	92.62	76.57	86.67
	μ_{th}	-	-	-	-	-	-	-	0.45	0.499983	0.499985
Parkinson's	sensitivity	96.55	96.55	**100**	89.66	93.1	75.86	96.55	**100**	89.66	**100**
	specificity	55.56	77.78	77.78	88.89	44.44	**100**	88.89	0	100	**100**
	Gmean	73.24	86.66	88.19	89.27	64.33	87.1	92.64	0	94.69	**100**
	F1-score	91.8	94.92	96.67	92.86	88.52	86.27	96.55	86.57	94.55	**100**
	μ_{th}	-		-	-	-	-	-	0.45	0.265354	0.266147

dant on how BLSMOTE performs, thus other oversampling methods that could provide better results may be explored. The distance-based clustering algorithm used in the proposed methods may introduce the curse of dimensionality, which could have also affected the results in the large data experiments. This issue may be addressed with other soft clustering algorithms that showed less dependency on similarity measure such as ones used in Ref. 44, 45. Alternatively, projecting data onto a lower-dimensional space using a technique such as Principle Components Analysis may be considered. Moreover, the performance of the proposed methods were observed to be more consistent on the simulated datasets than on the real-world datasets. This could be partly due to the difference in data uniformity. Thus, another potential improvement is to also factor in other information such as class density and local data density, which could be done using the techniques introduced in Ref. 46. Finally, the problem of small disjuncts in the minority class could be tackled by an adaptive selection on the number of clusters.

Bibliography

1. S. M. Erfani, S. Rajasegarar, S. Karunasekera and C. Leckie, High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning, *Pattern Recognit.* **58** (2016) 121–134.
2. V. Santucci, A. Milani and F. Caraffini, An optimisation-driven prediction method for automated diagnosis and prognosis, *Mathematics* **7**(11) (2019) p. 1051.
3. C. R. Qi, H. Su, M. Niessner, A. Dai, M. Yan and L. J. Guibas, Volumetric and multi-view cnns for object classification on 3d data, *IEEE Conf. Comput. Vision Pattern Recognit.*, (IEEE, 2016).
4. R. Moodley, F. Chiclana, F. Caraffini and J. Carter, Application of uninorms to market basket analysis, *Int. J. Intell. Syst.* **34**(1) (2019) 39–49.
5. M. H. Rafiei and H. Adeli, A new neural dynamic classification algorithm, *IEEE Trans. Neural Netw. Learn. Syst.* **28**(12) (2017) 3074–3083.
6. N. V. Chawla, K. W. Bowyer, L. O. Hall and W. P. Kegelmeyer, Smote: synthetic minority oversampling technique, *J. Artif. Intell. Res.* **16** (2002) 321–357.
7. C. Bunkhumpornpat, K. Sinapiromsaran and C. Lursinsap, Dbsmote: density-based synthetic minority over-sampling technique, *Appl.Intell.* **36**(3) (2012) 664–684.
8. W.-C. Lin, C.-F. Tsai, Y.-H. Hu and J.-S. Jhang, Clustering-based undersampling in class-imbalanced data, *Inf. Sci.* **409** (2017) 17–26.
9. H. K. Lee and S. B. Kim, An overlap-sensitive margin classifier for imbalanced and overlapping data, *Expert Syst. Appl.* (2018).
10. S. Das, S. Datta and B. B. Chaudhuri, Handling data irregularities in classification: Foundations, trends, and future challenges, *Pattern Recognit.* **81** (2018) 674–693.
11. J. Stefanowski, Overlapping, rare examples and class decomposition in learning classifiers from imbalanced data, *Emerg. Parad. Mach. Learn.*, (Springer, 2013), pp. 277–306.
12. P. Vuttipittayamongkol and E. Elyan, Neighbourhood-based undersampling approach for handling imbalanced and overlapped data, *Inf. Sci.* **509** (2020) 47–70.
13. C. Bunkhumpornpat and K. Sinapiromsaran, Dbmute: density-based majority under-sampling technique, *Knowl. Inf. Syst.* **50**(3) (2017) 827–850.
14. P. Vuttipittayamongkol, E. Elyan, A. Petrovski and C. Jayne, Overlap-based undersampling for improving imbalanced data classification, *Int. Conf. Intell. Data Eng. Autom. Learn.*, (Springer, 2018), pp. 689–697.
15. I. Triguero, M. Galar, S. Vluymans, C. Cornelis, H. Bustince, F. Herrera and Y. Saeys, Evolutionary undersampling for imbalanced big data classification, *IEEE Congr. Evol. Comput.*, (IEEE, 2015), pp. 715–722.
16. H. Han, W.-Y. Wang and B.-H. Mao, Borderline-smote: a new over-sampling method in imbalanced data sets learning, *Adv. Intell. Comput.* (2005) 878–887.
17. C. Bunkhumpornpat, K. Sinapiromsaran and C. Lursinsap, Safe-level-smote: Safe-level-synthetic

minority over-sampling technique for handling the class imbalanced problem, *Adv. Knowl. Discovery Data Min.* (2009) 475–482.
18. J. A. Sáez, J. Luengo, J. Stefanowski and F. Herrera, Smote–ipf: Addressing the noisy and borderline examples problem in imbalanced classification by a resampling method with filtering, *Inf. Sci.* **291** (2015) 184–203.
19. C. Jian, J. Gao and Y. Ao, A new sampling method for classifying imbalanced data based on support vector machine ensemble, *Neurocomputing* **193** (2016) 115–122.
20. H. He, Y. Bai, E. A. Garcia and S. Li, Adasyn: Adaptive synthetic sampling approach for imbalanced learning, *IEEE Int. Joint Conf. Neural Networks*, (IEEE, 2008), pp. 1322–1328.
21. D. Devi, B. Purkayastha *et al.*, Redundancy-driven modified tomek-link based undersampling: A solution to class imbalance, *Pattern Recognit. Lett.* **93** (2017) 3–12.
22. L. Nanni, C. Fantozzi and N. Lazzarini, Coupling different methods for overcoming the class imbalance problem, *Neurocomputing* **158** (2015) 48–61.
23. C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse and A. Napolitano, Rusboost: A hybrid approach to alleviating class imbalance, *IEEE Trans. Syst. Man Cybern. A Syst. Hum.* **40**(1) (2010) 185–197.
24. E. R. Fernandes and A. C. de Carvalho, Evolutionary inversion of class distribution in overlapping areas for multi-class imbalanced learning, *Inf. Sci.* **494** (2019) 141–154.
25. J. C. Bezdek, R. Ehrlich and W. Full, Fcm: The fuzzy c-means clustering algorithm, *Comput. Geosci.* **10**(2-3) (1984) 191–203.
26. G. E. Batista, R. C. Prati and M. C. Monard, A study of the behavior of several methods for balancing machine learning training data, *ACM Sigkdd Explorations Newsletter* **6**(1) (2004) 20–29.
27. S. Wang and X. Yao, Diversity analysis on imbalanced data sets by using ensemble models, *2009 IEEE Symp. Comput. Intell. Data Min.*, (IEEE, 2009), pp. 324–331.
28. H. Sun and S. Wang, Measuring the component overlapping in the gaussian mixture model, *Data Min. Knowl. Discovery* **23**(3) (2011) 479–502.
29. D. Dua and C. Graff, UCI machine learning repository (2017), `http://archive.ics.uci.edu/ml`.
30. J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez and F. Herrera, Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework., *J. Mult-Valued Log. S.* **17** (2011).
31. Y. LeCun and C. Cortes, MNIST handwritten digit database (2010) `http://yann.lecun.com/exdb/mnist`.
32. A. Ali-Gombe and E. Elyan, Mfc-gan: class-imbalanced dataset classification using multiple fake class generative adversarial network, *Neurocomputing* **361** (2019) 212–221.
33. M. Galar, A. Fernandez, E. Barrenechea, H. Bustince and F. Herrera, A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches, *IEEE Trans. Syst. Man Cybern. C Appl. Rev.* **42**(4) (2012) 463–484.
34. K. Napierala and J. Stefanowski, Types of minority class examples and their influence on learning classifiers from imbalanced data, *J. Intell. Inf. Syst.* **46**(3) (2016) 563–597.
35. U. R. Acharya, Y. Hagiwara and H. Adeli, Automated seizure prediction, *Epilepsy & Behavior* **88** (2018) 251–261.
36. C. Sun, H. Cui, W. Zhou, W. Nie, X. Wang and Q. Yuan, Epileptic seizure detection with eeg textural features and imbalanced classification based on easyensemble learning., *Int. J. Neural Syst.* (2019).
37. Y. Li, Z. Yu, Y. Chen, C. Yang, Y. Li, L. X. Allen and B. Li, Automatic seizure detection using fully convolutional nested lstm., *Int. J. Neural Syst.* (2020).
38. U. R. Acharya, S. L. Oh, Y. Hagiwara, J. H. Tan and H. Adeli, Deep convolutional neural network for the automated detection and diagnosis of seizure using eeg signals, *Comput. Biol. Med.* **100** (2018) 270–278.
39. N. Khoury, F. Attal, Y. Amirat, L. Oukhellou and S. Mohammed, Data-driven based approach to aid parkinsons disease diagnosis, *Sensors* **19**(2) (2019) p. 242.
40. S. Bhat, U. R. Acharya, Y. Hagiwara, N. Dadmehr and H. Adeli, Parkinson's disease: Cause factors, measurable indicators, and early diagnosis, *Comput. Biol. Med.* **102** (2018) 234–241.
41. P. Gómez-Vilda, Z. Galaz, J. Mekyska, J. M. F. Vicente, A. Gómez-Rodellar, D. Palacios-Alonso, Z. Smekal, I. Eliasova, M. Kostalova and I. Rektorova, Vowel articulation dynamic stability related to parkinsons disease rating features: Male dataset, *Int. J. Neural Syst.* **29**(02) (2019) p. 1850037.
42. A. H. Ansari, P. J. Cherian, A. Caicedo, G. Naulaers, M. De Vos and S. Van Huffel, Neonatal seizure detection using deep convolutional neural networks, *Int. J. Neural Syst.* **29**(04) (2019) p. 1850011.
43. F. Segovia, J. M. Gorriz Sáez, J. Ramírez Pérez De Inestrosa, F. J. Martínez Murcia, D. Castillo Barnes *et al.*, Assisted diagnosis of parkinsonism based on the striatal morphology, *Int. J. Neural Syst.* (2019).
44. J. Zhou, L. Chen, C. P. Chen, Y. Zhang and H.-X. Li, Fuzzy clustering with the entropy of attribute weights, *Neurocomputing* **198** (2016) 125–134.
45. H. Xia, J. Zhuang and D. Yu, Novel soft subspace clustering with multi-objective evolutionary approach for high-dimensional data, *Pattern Recognit.* **46**(9) (2013) 2562–2575.
46. M. Ahmadlou and H. Adeli, Enhanced probabilistic neural network with local decision circles: A robust classifier, *Integr. Comput.-Aid Eng.* **17**(3) (2010) 197–210.